日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００１年　２月２１日

出願番号
Application Number: 特願２００１－０４５０７２

[ST.10/C]: [ＪＰ２００１－０４５０７２]

出願人
Applicant(s): 株式会社半導体理工学研究センター

CERTIFIED COPY OF
PRIORITY DOCUMENT

２００２年　２月１９日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造

出証番号　出証特２００２－３００８６１９

【書類名】　特許願
【整理番号】　172782
【提出日】　平成13年 2月21日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 12/08　310
　　G06F 15/16　645
【発明者】
　【住所又は居所】　東京都港区赤坂７－５－５１　富山県赤坂会館
　【氏名】　酒井　敦
【発明者】
　【住所又は居所】　神奈川県横浜市港北区日吉７－１６－２７－１０３
　【氏名】　天野　英晴
【特許出願人】
　【識別番号】　396023993
　【住所又は居所】　東京都港区新橋６丁目１６番１０号
　【氏名又は名称】　株式会社半導体理工学研究センター
【代理人】
　【識別番号】　100062144
　【弁理士】
　【氏名又は名称】　青山　葆
【選任した代理人】
　【識別番号】　100086405
　【弁理士】
　【氏名又は名称】　河宮　治
【手数料の表示】
　【予納台帳番号】　013262
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1

【物件名】　　図面　1

【物件名】　　要約書　1

【包括委任状番号】　9608010

【プルーフの要否】　　要

【書類名】　明細書

【発明の名称】　キャッシュメモリシステム装置

【特許請求の範囲】

【請求項１】　プロセッサとメインメモリとの間に高速アクセスが可能な小容量のキャッシュメモリを設けてなるキャッシュメモリシステム装置において、

あらかじめプログラムされたソフトウェアにしたがって上記キャッシュメモリに対するデータ転送制御であるソフトウェア制御を行うソフトキャッシュ制御部と、

所定のハードウェアを用いて上記キャッシュメモリに対するデータ転送制御であるハードウェア制御を行うハードキャッシュ制御部と、

を備え、

上記プロセッサは、ソフトキャッシュ制御部に対してソフトウェア制御を実行させ、該ソフトウェア制御が実行不可能になると、ハードキャッシュ制御部に対してハードウェア制御を行わせることを特徴とするキャッシュメモリシステム装置。

【請求項２】　上記プロセッサは、ソフトウェア制御時にキャッシュミスが発生すると、上記ハードキャッシュ制御部に対してハードウェア制御を行わせることを特徴とする請求項１記載のキャッシュメモリシステム装置。

【請求項３】　上記ソフトキャッシュ制御部は、コンパイラの静的予測によって生成されたコードにしたがって、所望のデータをキャッシュメモリ内に格納することを特徴とする請求項１又は２記載のキャッシュメモリシステム装置。

【請求項４】　上記ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行する前に、該データ読み出し命令で指定されるメインメモリのアドレスのデータを読み出してキャッシュメモリに格納することを特徴とする請求項３記載のキャッシュメモリシステム装置。

【請求項５】　上記ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行すると同時に該データ読み出し命令で指定されたメインメモリのアドレスのデータをキャッシュメモリから

プロセッサに転送させることを特徴とする請求項４記載のキャッシュメモリシステム装置。

【請求項６】　上記ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行する前に、キャッシュメモリに対してプロセッサからのデータを格納するアドレスを指定することを特徴とする請求項３、４又は５記載のキャッシュメモリシステム装置。

【請求項７】　上記ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行し、指定したキャッシュメモリのアドレスに書き込まれたプロセッサからのデータを、該データ書き込み命令で指定されたメインメモリのアドレスに書き込むことを特徴とする請求項６記載のキャッシュメモリシステム装置。

【請求項８】　上記ハードキャッシュ制御部は、複数ウェイのセットアソシエイティブ方式を用いてキャッシュメモリのライン管理を行い、上記ソフトキャッシュ制御部は、該複数ウェイの内少なくとも１ウェイをフルアソシエイティブ方式を用いてキャッシュメモリのライン管理を行うことを特徴とする請求項１、２、３、４、５、６又は７記載のキャッシュメモリシステム装置。

【請求項９】　上記ソフトキャッシュ制御部は、キャッシュメモリに対するデータ転送制御を行う転送制御用プロセッサで構成されることを特徴とする請求項１、２、３、４、５、６、７又は８記載のキャッシュメモリシステム装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、プロセッサとメインメモリとの間に高速アクセスが可能な小容量のキャッシュメモリを設けたキャッシュメモリシステム装置に関し、特に複数のプロセッサに対して無同期実行を行うマルチプロセッサシステムに使用するキャッシュメモリシステム装置に関する。

【０００２】

【従来の技術】

図１４は、従来のキャッシュメモリシステム装置の構成例を示した概略のブロ

ック図である。

図１４において、キャッシュメモリシステム装置１００は、プロセッサ１０１が、キャッシュメモリ部１０２を介してメインメモリ１０３に接続されて構成されている。更に、キャッシュメモリ部１０２は、タグメモリ１０５と、キャッシュメモリ１０６、タグメモリ１０５に格納されているタグの対応表を参照してキャッシュメモリ１０６に対するデータ転送制御を行うキャッシュ制御部１０７で構成されている。

【０００３】

一方、キャッシュメモリシステム装置１００では、キャッシュメモリ１０６にデータがあるか否か（ヒットするか否か）によって、アクセス時間が変わる。このため、キャッシュ制御部１０７は、キャッシュヒット率を向上させるために、あらかじめキャッシュメモリ１０６にデータを準備するプリフェッチ機構を有する場合がある。このようなキャッシュメモリシステム装置１００において、プロセッサ１０１は、キャッシュメモリ１０６にアクセスするデータが存在する場合はキャッシュメモリ１０６から、キャッシュメモリ１０６にアクセスするデータが存在しない場合はメインメモリ１０３からデータが供給される。

【０００４】

【発明が解決しようとする課題】

しかし、このような構成では、静的スケジューリング技術によってプリフェッチ機構を使用したとしても、キャッシュのヒット率を１００％にすることは不可能であった。このため、マルチプロセッサ構成で無同期実行を行うマルチプロセッサシステム装置では、図１４で示したような従来のキャッシュメモリシステム装置を使用して無同期実行を行うことは困難であった。

【０００５】

本発明は、上記のような問題を解決するためになされたものであり、データ転送を管理するプロセッサを追加すると共にソフトウェアで制御される動作モードを追加して、コンパイラにキャッシュメモリのライン情報を管理する機構を設けることによって、キャッシュミスが発生せずに無同期実行が可能となるキャッシュメモリシステム装置を得ることを目的とする。

【０００６】

【課題を解決するための手段】

この発明に係るキャッシュメモリシステム装置は、プロセッサとメインメモリとの間に高速アクセスが可能な小容量のキャッシュメモリを設けてなるキャッシュメモリシステム装置において、あらかじめプログラムされたソフトウェアに従ってキャッシュメモリに対するデータ転送制御であるソフトウェア制御を行うソフトキャッシュ制御部と、所定のハードウェアを用いてキャッシュメモリに対するデータ転送制御であるハードウェア制御を行うハードキャッシュ制御部とを備え、プロセッサは、ソフトキャッシュ制御部に対してソフトウェア制御を実行させ、該ソフトウェア制御が実行不可能になると、ハードキャッシュ制御部に対してハードウェア制御を行わせるものである。

【０００７】

具体的には、上記プロセッサは、ソフトウェア制御時にキャッシュミスが発生すると、ハードキャッシュ制御部に対してハードウェア制御を行わせるようにした。

【０００８】

また、上記ソフトキャッシュ制御部は、コンパイラの静的予測によって生成されたコードにしたがって、所望のデータをキャッシュメモリ内に格納するようにしてもよい。

【０００９】

具体的には、上記ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行する前に、該データ読み出し命令で指定されるメインメモリのアドレスのデータを読み出してキャッシュメモリに格納するようにした。

【００１０】

更に、上記ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行すると同時に該データ読み出し命令で指定されたメインメモリのアドレスのデータをキャッシュメモリからプロセッサに転送させるようにしてもよい。

【００１１】

一方、上記ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行する前に、キャッシュメモリに対してプロセッサからのデータを格納するアドレスを指定するようにした。

【００１２】

更に、上記ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行し、指定したキャッシュメモリのアドレスに書き込まれたプロセッサからのデータを、該データ書き込み命令で指定されたメインメモリのアドレスに書き込むようにしてもよい。

【００１３】

また、上記ハードキャッシュ制御部は、複数ウェイのセットアソシエイティブ方式を用いてキャッシュメモリのライン管理を行い、上記ソフトキャッシュ制御部は、該複数ウェイの内少なくとも１ウェイをフルアソシエイティブ方式を用いてキャッシュメモリのライン管理を行うようにした。

【００１４】

また具体的には、上記ソフトキャッシュ制御部は、キャッシュメモリに対するデータ転送制御を行う転送制御用プロセッサで構成されるようにした。

【００１５】

【発明の実施の形態】

次に、図面に示す実施の形態に基づいて、本発明を詳細に説明する。

図１は、本発明の実施の形態におけるキャッシュメモリシステム装置を使用するマルチプロセッサシステムの例を示した概略のブロック図である。

図１において、マルチプロセッサシステムは、複数のプロセッサＣＰＵ１～ＣＰＵｎ（ｎは、ｎ＞１の自然数）と、該各プロセッサＣＰＵ１～ＣＰＵｎに対応して設けられたキャッシュメモリ部ＣＭ１～ＣＭｎと、集中共有メモリ（centralized shared memory）ＣＳＭと、データ交信用バス３とを備えている。各プロセッサＣＰＵ１～ＣＰＵｎは、対応するキャッシュメモリ部ＣＭ１～ＣＭｎに接続され、各キャッシュメモリ部ＣＭ１～ＣＭｎは、データ交信用バス３を介して互いに接続されると共に集中共有メモリＣＳＭに接続されている。

【００１６】

なお、このような構成において、各プロセッサＣＰＵ１～ＣＰＵｎの交信には、各プロセッサのレジスタ間で交信するプロセッサ間通信というシステムを使用しているが、交信データが多くレジスタ間通信では処理しきれない場合は、キャッシュメモリ部ＣＭ１～ＣＭｎと集中共有メモリＣＳＭとの間に、それぞれ交信用のローカルメモリＣＯＭＭ１～ＣＯＭＭｎを対応させて設けるようにしてもよい。この場合、各プロセッサＣＰＵ１～ＣＰＵｎは、指定するアドレスによってローカルメモリＣＯＭＭ１～ＣＯＭＭｎ又は集中共有メモリＣＳＭにアクセスすることができる。

【００１７】

図２は、本発明の実施の形態におけるキャッシュメモリシステム装置の例を示した概略のブロック図であり、図２では、図１で示した任意の１つのプロセッサＣＰＵｍ（ｍ＝１～ｎ）の場合を例にして示している。

図２において、キャッシュメモリシステム装置１は、プロセッサＣＰＵｍと、キャッシュメモリ部ＣＭｍと、メインメモリをなす集中共有メモリＣＳＭとで構成されている。

【００１８】

また、キャッシュメモリ部ＣＭｍは、リードバッファ５を有するキャッシュメモリ６と、タグメモリ７と、ソフトウェア制御時に該キャッシュメモリ６に対するデータ転送制御を行うＤＴＣ（Data Transfer Controller）８と、ハードウェア制御時に該キャッシュメモリ６に対するデータ転送制御を行うハードキャッシュ制御部９とで構成されている。ＤＴＣ８は、ソフトウェア制御時にプロセッサＣＰＵｍから入力されるアドレスデータのチェックを行うアドレスチェック機構を備え、キャッシュメモリ６に対するデータ転送の管理を行うプロセッサで構成され、ソフトキャッシュ制御部をなすものである。

【００１９】

集中共有メモリＣＳＭのアドレスは、タグ（tag）部とキー（key）部に分けられ、タグメモリ７は、キャッシュメモリ６のどの場所に集中共有メモリＣＳＭのどのアドレスのデータが格納されているかの対応を示すタグとキーの対応表を格

納する。ソフトウェア制御時にはＤＴＣ８が、ハードウェア制御時にはハードキャッシュ制御部９が、タグメモリ７に格納されている対応表のデータ管理を行う。このことから、ＤＴＣ８及びハードキャッシュ制御部９は、タグメモリ７の対応表のタグをチェックすることによって、プロセッサＣＰＵｍから要求された集中共有メモリＣＳＭのアドレスのデータがキャッシュメモリ６内にあるか否かが分かる。

【００２０】

ハードキャッシュ制御部９は、プロセッサＣＰＵｍから要求された集中共有メモリＣＳＭのアドレスのデータがキャッシュメモリ６内にあった場合、キャッシュメモリ６のどのアドレスに格納されているかがタグから分かるため、キャッシュメモリ６に該アドレスを指定してプロセッサＣＰＵｍに読み出すようにする。また、ハードキャッシュ制御部９は、プロセッサＣＰＵｍから要求された集中共有メモリＣＳＭのアドレスのデータがキャッシュメモリ６内になかった場合、プロセッサＣＰＵｍから要求されたデータを集中共有メモリＣＳＭからキャッシュメモリ６へ読み出す。このとき、ハードキャッシュ制御部９は、集中共有メモリＣＳＭからのデータをキャッシュメモリ６のどのアドレスに格納するかを決定し、キャッシュメモリ６に該決定したアドレスを指定することによって、該アドレスに集中共有メモリＣＳＭから読み出されたデータが格納される。

【００２１】

ここで、ソフトウェア制御とは、コンパイラの静的予測によって生成されたコードにしたがってデータをキャッシュメモリ６上に準備する動作モードであり、ある一定区間内のキャッシュのヒット率を１００％にして、マルチプロセッサ構成時に無同期実行を可能にする。これに対してハードウェア制御とは、従来行われている制御であり、プロセッサＣＰＵｍからの要求があって初めてキャッシュメモリ６に対して、マッピング、キャッシュヒットの確認、キャッシュメモリ６からのデータの追い出し、及び集中共有メモリＣＳＭからキャッシュメモリ６へのデータ転送が行われる動作モードである。

【００２２】

なお、ハードウェア制御では、キャッシュメモリ６のどの場所にどのデータを

格納するか、どのデータをキャッシュメモリ６から追い出すか等はすべてハードウェアで管理されることになるが、今後の挙動を把握しているわけではない。このことから、過去のアクセスパターンを利用した何らかの法則に基づいて配置を決める。例えば、格納場所に関しては４ウェイ・セットアソシエイティブ（4-way set-associative）等の方法で、データ追い出しの管理に関してはＦＩＦＯ、ＬＲＵ等のハードウェアで実現できる範囲の方法で行う。

【００２３】

図３は、４ウェイ・セットアソシエイティブ方式を用いた場合のタグメモリ７に格納されているタグの対応表の例を示した図であり、図３では、８種類のタグを有する場合を例にして示している。

図３において、タグ０には８の倍数のキーが、タグ１には「８の倍数＋１」のキーが第１～第４ウェイ（ｗａｙ）の各ウェイごとにそれぞれ対応付けされているように、タグｎ（ｎ＝０～７）には「８の倍数＋ｎ」のキーが各ウェイ（ｗａｙ）ごとに対応付けされている。このように、タグに対するキーを何らかの法則に基づいて各ウェイごとに配置されており、ハードウェア制御時には、ハードキャッシュ制御部９は、このような４ウェイ・セットアソシエイティブ方式を用いてキャッシュメモリ６のライン管理を行う。

【００２４】

一方、ＤＴＣ８は、プロセッサＣＰＵｍとキャッシュメモリ６との間、及びキャッシュメモリ６と集中共有メモリＣＳＭとの間のそれぞれのデータ転送を、コンパイラがあらかじめ生成した命令列にしたがって制御するプロセッサである。ＤＴＣ８とコンパイラのデータ解析を組み合わせることで、所望のデータをあらかじめキャッシュメモリ６に用意して、キャッシュのヒット率を１００％にすることができる。更に、ＤＴＣ８は、ソフトウェア制御時に、少なくとも１ウェイのみフルアソシエイティブ（full-associative）方式を用い、他のウェイはセットアソシエイティブ方式を用いてキャッシュメモリ６のライン管理を行う。

【００２５】

図４は、図３で示したタグの対応表のソフトウェア制御時における例を示した図である。なお、図４では、４ウェイの内、第４ウェイのみにフルアソシエイテ

ィブ方式を用いる場合を例にして示している。

図４において、第１～第３ウェイは、ハードウェア制御時と同様にセットアソシエイティブ方式を用い、第４ウェイのみフルアソシエイティブ方式を用いてキャッシュメモリ６のライン管理が行われる。

【００２６】

何らかの法則に基づいて配置を決めるセットアソシエイティブ方式に対して、フルアソシエイティブ方式では、使用されていないキャッシュメモリ６のアドレスに自由に配置することができる。このため、セットアソシエイティブ方式の領域でデータの衝突が発生し、データを排除しなければ新しいデータが読み込めないような場合、ＤＴＣ８は、新しいデータをフルアソシエイティブ方式の領域の使用されていないアドレスに読み込むことによってキャッシュの利用効率を高めることができる。

【００２７】

一方、ソフトウェア制御からハードウェア制御に移行した場合、図５で示すように、ハードキャッシュ制御部９によって、フルアソシエイティブ方式の領域でセットアソシエイティブ方式の法則に基づかないタグのキーは削除されて、タグの対応表の更新が行われる。

【００２８】

次に、コンパイラによって生成されるＤＴＣ８への命令について説明する。ＤＴＣ８への命令としては、キャッシュメモリ６とプロセッサＣＰＵｍとの仲介を行う際に発行される「ＬＡＣ」，「ＳＡＣ」と、キャッシュメモリ６と集中共有メモリＣＳＭとの仲介を行う際に発行される「ＰＬ（PreLoad）」，「ＰＳ（PostStore）」がある。ＬＡＣ命令は、キャッシュメモリ６の特定部分のデータをプロセッサＣＰＵｍに読み出させる命令であり、ＳＡＣ命令は、キャッシュメモリ６の特定の部分にデータを書き込ませる命令である。また、ＰＬ命令は、集中共有メモリＣＳＭからキャッシュメモリ６の特定部分にデータを読み出させる命令であり、ＰＳ命令は、キャッシュメモリ６の特定部分のデータを集中共有メモリＣＳＭに書き込ませる命令である。

【００２９】

ソフトウェア制御は、コンパイラによるデータアクセスの静的解析が完全に可能な部分で行われることから、該静的解析を生かしてコンパイル時にキャッシュメモリ６まで制御を行うことができ、プレロード（PreLoad）等も行うことができる。このため、コンパイラがキャッシュメモリ６の制御を行った場合、当然キャッシュメモリ６のアドレスマップはコンパイラ内で管理される。したがって、実際の動作時にはハードウェアにアドレスマップがないため、プロセッサＣＰＵｍがロード（Load）命令で集中共有メモリＣＳＭのアドレスを指定しても、該アドレスのデータがキャッシュメモリ６のどの場所に格納されているかは不明となる。これを解決するために、ソフトウェア制御では、キャッシュメモリ６のアドレスを指定したＬＡＣ命令及びＳＡＣ命令を使用して、ロード及びストア（Store）を行う。

【００３０】

ＤＴＣ８が実行する命令のフォーマットとしては、「ＬＡＣ cache-addr」、「ＳＡＣ cache-addr」、「ＰＬ cache-addr mem-addr」及び「ＰＳ cache-addr mem-addr」となる。なお、「cache-addr」には、キャッシュメモリ６のアドレスが、「mem-addr」には、集中共有メモリＣＳＭのアドレスが示される。「ＬＡＣ cache-addr」は、「cache-addr」で示されたキャッシュメモリ６のアドレスのデータを読み出してリードバッファ５に一時的に格納させ、プロセッサＣＰＵｍにロード命令が発行されてから１クロックでプロセッサＣＰＵｍに出力できるようにする命令である。「ＬＡＣ cache-addr」は、「cache-addr」に示されたキャッシュメモリ６のアドレスのデータを読み出すロード命令がプロセッサＣＰＵｍに発行される１クロック前に、ＤＴＣ８に発行される。

【００３１】

「ＳＡＣ cache-addr」は、「cache-addr」で示されたキャッシュメモリ６のアドレスにプロセッサＣＰＵｍからのデータを格納させる命令である。「ＳＡＣ cache-addr」は、「cache-addr」で示されたキャッシュメモリ６のアドレスにデータを書き込むストア命令がプロセッサＣＰＵｍに発行される１クロック前に、ＤＴＣ８に発行される。

【００３２】

「ＰＬ cache-addr mem-addr」は、「mem-addr」で示された集中共有メモリＣＳＭのアドレスのデータを読み出して、「cache-addr」で示されたキャッシュメモリ６のアドレスに書き込ませる命令である。「ＰＬ cache-addr mem-addr」は、「cache-addr」に示されたキャッシュメモリ６のアドレスのデータを読み出すロード命令がプロセッサＣＰＵｍに発行されるよりも、少なくとも集中共有メモリＣＳＭからデータを読み出してキャッシュメモリ６に書き込まれるのに要するクロック数だけ前にＤＴＣ８に発行される。

【００３３】

「ＰＳ cache-addr mem-addr」は、「cache-addr」で示されたキャッシュメモリ６のアドレスのデータを読み出して、「mem-addr」で示された集中共有メモリＣＳＭのアドレスに書き込ませる命令である。「ＰＳ cache-addr mem-addr」は、「cache-addr」に示されたキャッシュメモリ６のアドレスのデータを集中共有メモリＣＳＭに書き込ませるストア命令がプロセッサＣＰＵｍに発行された直後、例えばプロセッサＣＰＵｍに該ストア命令が発行される１クロック後に、ＤＴＣ８に発行される。

【００３４】

このように、ＤＴＣ８は、コンパイラによってスケジューリングされた命令にしたがって動作し、ソフトウェア制御では、キャッシュメモリ６内のどのアドレスにデータを格納するかというところまで管理を行う。このことから、意図しないキャッシュメモリ６内のデータの追い出しを防止することができ、不必要なデータの消去等ができることから、キャッシュのヒット率を１００％にすることができる。更に、プロセッサＣＰＵｍの集中共有メモリＣＳＭへのアクセスが高速になるだけでなく、集中共有メモリＣＳＭへのアクセスが安定することによって、クロックレベルでのプロセッサＣＰＵｍの動作の予測が可能となり、マルチプロセッサシステム装置で使用した場合に、各プロセッサによる無同期実行等を実現することができる。

【００３５】

次に、ソフトウェア制御時のプロセッサＣＰＵｍ及びＤＴＣ８の動作について説明する。

図６は、プレロード実行時における各部の動作例を示したブロック図であり、図６を用いてプレロード実行時における各部の動作を説明する。なお、点線の矢印が制御の流れを、実線の矢印がデータの流れを示しており、ソフトウェア制御時の動作説明に関係しない部分は省略している。

【００３６】

図６において、プレロードを実行する場合、まずＤＴＣ８は、ＰＬ命令が発行され、該ＰＬ命令で指定されたキャッシュメモリ６のアドレスに有効なデータが格納されていないことをチェックし、有効なデータが格納されている場合は、所定のミス信号ｍｉｓｓをプロセッサＣＰＵｍに出力する。プロセッサＣＰＵｍは、ミス信号ｍｉｓｓが入力されると、ソフトウェア制御からハードキャッシュ制御部９を使用するハードウェア制御に切り替える。また、有効なデータが格納されていない場合、ＤＴＣ８は、該ＰＬ命令で指定された集中共有メモリＣＳＭのアドレスのデータを読み出して、該ＰＬ命令で指定されたキャッシュメモリ６のアドレスに格納させる。

【００３７】

次に、ＤＴＣ８は、ＬＡＣ命令が発行され、該ＬＡＣ命令で指定されたキャッシュメモリ６のアドレスのデータを読み出して、リードバッファ５に格納させる。次に、プロセッサＣＰＵｍは、ロード命令が発行され、該ロード命令で指定された集中共有メモリＣＳＭのアドレスをＤＴＣ８に出力する。ＤＴＣ８は、入力されたアドレスが上記ＰＬ命令で指定された集中共有メモリＣＳＭのアドレスと一致するか否かをチェックし、一致した場合は、キャッシュヒットしたことを示す所定のヒット信号ｈｉｔをプロセッサＣＰＵｍ及びリードバッファ５にそれぞれ出力する。ヒット信号ｈｉｔが入力されたリードバッファ５は、格納しているデータをプロセッサＣＰＵｍに出力する。このようにして、ＤＴＣ８によるプレロードが行われる。

【００３８】

一方、ＤＴＣ８は、入力されたアドレスが上記ＰＬ命令で指定された集中共有メモリＣＳＭのアドレスと一致しなかった場合は、キャッシュミスしたことを示す所定のミス信号ｍｉｓｓをプロセッサＣＰＵｍに出力する。プロセッサＣＰＵ

ｍは、ミス信号ｍｉｓｓが入力されると、ソフトウェア制御からハードキャッシュ制御部９を使用するハードウェア制御に切り替える。

【００３９】

図７は、ハードウェア制御時においてロード命令を実行しキャッシュミスした場合の各部の動作例を示したブロック図である。

図７において、プロセッサＣＰＵｍは、ロード命令が発行され、該ロード命令で指定された集中共有メモリＣＳＭのアドレスをハードキャッシュ制御部９に出力する。ここで、タグメモリ７には、キャッシュメモリ６のどの場所に集中共有メモリＣＳＭのどのアドレスのデータが格納されているかを示す対応表であるタグが格納されている。ハードキャッシュ制御部９は、ハードウェア制御時において該タグメモリ７に格納されているタグの内容を管理している。

【００４０】

ハードキャッシュ制御部９は、タグをチェックしてプロセッサＣＰＵｍから入力された集中共有メモリＣＳＭのアドレスのデータがキャッシュメモリ６内にあるか否かをチェックする。キャッシュメモリ６内にプロセッサＣＰＵｍから指定されたアドレスのデータがなかった場合、ハードキャッシュ制御部９は、キャッシュミスしたことを示す所定のミス信号ｍｉｓｓをプロセッサＣＰＵｍに出力する。次に、ハードキャッシュ制御部９は、プロセッサＣＰＵｍから指定されたアドレスのデータを集中共有メモリＣＳＭから読み出してキャッシュメモリ６に格納してタグメモリ７内のタグの対応表を更新する。更に、ハードキャッシュ制御部９は、更新した対応表に基づいて、プロセッサＣＰＵｍから指定されたアドレスのデータをキャッシュメモリ６から読み出してプロセッサＣＰＵｍに出力する。

【００４１】

一方、図８は、ハードウェア制御時においてロード命令を実行しキャッシュヒットした場合の各部の動作例を示したブロック図である。

図８において、プロセッサＣＰＵｍは、ロード命令が発行され、該ロード命令で指定された集中共有メモリＣＳＭのアドレスをハードキャッシュ制御部９に出力する。ハードキャッシュ制御部９は、タグをチェックしてプロセッサＣＰＵｍ

から入力された集中共有メモリＣＳＭのアドレスのデータがキャッシュメモリ６内にあるか否かをチェックする。

【００４２】

キャッシュメモリ６内にプロセッサＣＰＵｍから指定されたアドレスのデータがあった場合、ハードキャッシュ制御部９は、キャッシュヒットしたことを示す所定のヒット信号ｈｉｔをプロセッサＣＰＵｍに出力する。次に、ハードキャッシュ制御部９は、タグの対応表に基づいて、プロセッサＣＰＵｍから指定されたアドレスのデータをキャッシュメモリ６から読み出してプロセッサＣＰＵｍに出力する。このようにして、ハードウェア制御動作が行われる。

【００４３】

次に、図９は、ポストストア（PostStore）実行時における各部の動作例を示したブロック図であり、図９を用いてポストストア実行時における各部の動作を説明する。なお、図９においても、点線の矢印が制御の流れを、実線の矢印がデータの流れを示しており、ソフトウェア制御時の動作説明に関係しない部分は省略している。

図９において、ポストストアを実行する場合、まずＤＴＣ８は、ＳＡＣ命令が発行され、該ＳＡＣ命令で指定されたアドレスにデータを書き込むようキャッシュメモリ６の動作制御を行う。

【００４４】

次に、プロセッサＣＰＵｍは、ストア命令が発行され、該ストア命令で指定された集中共有メモリＣＳＭのアドレスをＤＴＣ８に出力すると共に、キャッシュメモリ６にデータを出力し、該データは、上記ＳＡＣ命令で指定されたアドレスに格納される。ＤＴＣ８は、キャッシュメモリ６に格納されたデータにおける集中共有メモリＣＳＭのアドレスをタグから求め、ストア命令で指定された集中共有メモリＣＳＭのアドレスと一致するかチェックする。ＤＴＣ８は、一致しなかった場合は、プロセッサＣＰＵｍに対して所定のミス信号ｍｉｓｓを出力し、一致した場合は、所定のヒット信号ｈｉｔを出力する。プロセッサＣＰＵｍは、ヒット信号ｈｉｔが入力されると引き続きソフトウェア制御で動作させ、ミス信号ｍｉｓｓが入力されると、ソフトウェア制御からハードキャッシュ制御部９を使

用するハードウェア制御に切り替える。

【００４５】

次に、ＤＴＣ８は、ＰＳ命令が発行され、該ＰＳ命令で指定されたキャッシュメモリ６のアドレスに格納されたデータが有効であるか否かをチェックし、データが有効であれば、該データを、指定された集中共有メモリＣＳＭに転送した後、無効化を行う。また、ＤＴＣ８は、データが無効であれば、所定のミス信号ｍｉｓｓをプロセッサＣＰＵｍに出力し、プロセッサＣＰＵｍは、ソフトウェア制御からハードキャッシュ制御部９を使用するハードウェア制御に切り替える。

【００４６】

このようにして、ＤＴＣ８によるポストストアが行われる。なお、上記プレロード及びポストストアにおいて、集中共有メモリＣＳＭにアクセスする必要がない場合、例えばデータがまだプロセッサで使用される場合等では、ＰＬ命令及びＰＳ命令を発行しないようにＤＴＣ８に対するプログラムコードが生成される。

【００４７】

プロセッサＣＰＵｍ及びＤＴＣ８に対する各命令の発行タイミングは、あらかじめコンパイラによってスケジューリングされる。以下、該スケジューリング方法について説明する。

図１０は、コンパイラによって行われるプロセッサＣＰＵｍに対するアセンブラコード及びＤＴＣ８に対するＤＴＣコードの生成過程を示したブロック図である。なお、図１０で示した各処理は、通常ワークステーション等のコンピュータによって実行されるコンパイラによって行われる。

【００４８】

図１０において、まず最初に、Ｃ言語又はＦＯＲＴＲＡＮ言語でプログラムされた逐次型のプログラムに対して自動並列化処理が行われ、プログラムを並列性に応じて粗粒度（Coarse-Grain）、中粒度（Medium-Grain）及び細粒度（Fine-Grain）といった３つの粒度に分けられる。

ここで、自動並列化処理について簡単に説明する。自動並列化処理では、まずプログラムを分岐部分ですべて区切り、分岐のない一連のプログラムに分け、これを粗粒度タスク又はタスクと呼ぶ。

【００４９】

粗粒度並列化では、まず該タスク間に依存がないか調べ、依存がなければ各タスクを並列に処理するようにする。次に、中粒度並列化では、上記タスク内を調べて更に簡単に並列化できるループ、すなわち各イタレーションで計算されるデータに依存がないループであるか否かを調べ、並列化できるループであれば並列化を行う。このような中粒度並列化は、別名ループ並列化ともいう。次に、細粒度並列化では、タスクがループでないか又は並列化できないループであった場合に適用される並列化手法であり、プログラムの各文（例えばａ＝ｂ＋ｃ等）の間の並列性を抽出する方法である。

【００５０】

次に、上記細粒度並列化処理を行った後のコードで、プロセッサＣＰＵｍ用の分岐のない一連の命令コードである細粒度コードを使用して、アセンブラのＣＰＵコードを生成し、該生成したＣＰＵコードに基づいてＤＴＣコードを生成する。一方、ＤＴＣ８をクロックレベルで厳密に動作させるためには、データのロード又はストア等で発生するネットワークパケットの衝突まで予測する必要がある。このことから、ネットワークレベルでのスケジューリングを行うことによって該予測を行い、完全にクロックレベルでの動作を保証したＤＴＣコードを作成する。

【００５１】

次に、上記細粒度コードから生成されたＣＰＵコードに基づいてＤＴＣコードを生成する際に実行される、コンパイラの静的予測に基づくソフトウェア制御時のＤＴＣ用命令の配置アルゴリズムについて説明する。

まず、コンパイラは、通常のコンパイラ技術によってアセンブラのＣＰＵコードの生成を行い、該生成したＣＰＵコードからデータをロードするロード命令、データを書き込むストア命令を探し、対応するＤＴＣ命令、すなわちＬＡＣ命令、ＳＡＣ命令、ＰＬ命令及びＰＳ命令を生成する。

【００５２】

次に、コンパイラは、ロード又はストアするデータが必ずキャッシュヒットするという前提条件のもとで、生成したＣＰＵコードをシミュレーションし、各命

令の正確なクロックを算出する。更に、コンパイラは、プロセッサＣＰＵｍがロード又はストアするときに確実にキャッシュメモリ６にデータがあるようにＤＴＣ用命令であるＬＡＣ命令、ＳＡＣ命令、ＰＬ命令及びＰＳ命令の発行タイミングの調整を行う。

【００５３】

ここで、コンパイラによって行われるＤＴＣ用命令の発行タイミングの決定方法について説明する。

プロセッサＣＰＵｍの実行命令には、実行順に０から昇順に実行クロックが決められており、ＤＴＣ命令列の実行クロックはプロセッサＣＰＵｍの命令列の実行クロックを基準とする。したがって、プロセッサＣＰＵｍで実行される最初の命令よりも前に実行するＤＴＣ命令は、実行クロックがマイナスになる。また、プロセッサＣＰＵｍ及びＤＴＣ８は、各クロックで実行することができる命令は共に１命令のみである。

【００５４】

コンパイラは、ＬＡＣ命令をプロセッサＣＰＵｍのロード命令の１クロック前に、ＳＡＣ命令をプロセッサＣＰＵｍのストア命令の１クロック後に必ず実行されるようにＤＴＣ命令列に配置し、該配置されたＬＡＣ命令及びＳＡＣ命令の移動を禁止する。次に、コンパイラは、ＰＬ命令を所定のアルゴリズムにしたがってＤＴＣ命令列に配置し、この後ＰＳ命令を所定のアルゴリズムにしたがってＤＴＣ命令列に配置する。

【００５５】

この際、ＤＴＣ８がＰＬ命令又はＰＳ命令を実行する際、集中共有メモリＣＳＭにアクセスするために数十クロックを要するため、コンパイラは、この間、集中共有メモリＣＳＭにアクセスさせる命令をＤＴＣ命令列に配置することができない。なお、以下、ＰＬ命令実行時における集中共有メモリＣＳＭへのアクセスに要するクロック範囲をロード範囲と呼び、ＰＳ命令実行時における集中共有メモリＣＳＭへのアクセスに要するクロック範囲をストア範囲と呼ぶ。

【００５６】

一方、コンパイラは、プロセッサＣＰＵｍにおけるすべてのロード命令及びス

トア命令に対して、ＰＬ命令及びＰＳ命令を対応付けするのではなく、データのライフタイムを算出し、最初のプロセッサＣＰＵｍによるロード時にＰＬ命令を、最後のプロセッサＣＰＵｍによるストア時にＰＳ命令を対応付けする。なお、これ以外の場合、キャッシュメモリ６にデータが格納されていることから、コンパイラは、ＤＴＣ８によってＬＡＣ命令及びＳＡＣ命令で処理されるようにする。

【００５７】

図１１は、コンパイラによって行われるＰＬ命令の配置アルゴリズムを示したフローチャートであり、図１１を用いて、ＰＬ命令の配置方法について説明する。なお、図１１の各フローで行われる処理は、特に明記しない限りコンパイラによって行われる。

図１１において、まず最初に、プロセッサＣＰＵｍのロード命令実行クロックから、集中共有メモリＣＳＭへのアクセスに要するクロック数を差し引いたクロックを各ＰＬ命令の仮の実行クロックに設定する（ステップＳ１）。次に、実行クロックの最も遅い、すなわち最後に実行されるＰＬ命令を対象にする（ステップＳ２）。

【００５８】

この後、対象となっているＰＬ命令が、すでにＤＴＣ命令列に配置されているＬＡＣ命令又はＳＡＣ命令と重なっているか否かを調べ（ステップＳ３）、重なっている場合（ＹＥＳ）は、対象となっているＰＬ命令をＬＡＣ命令及びＳＡＣ命令と重ならないクロックまで実行クロックを減少させる（ステップＳ４）。次に、現在ＤＴＣ命令列に配置されているＰＬ命令が、配置しようとしているＰＬ命令のロード範囲に重なるか否かを調べ（ステップＳ５）、重なる場合（ＹＥＳ）は、重なったＰＬ命令がロード範囲外になるまで対象となっているＰＬ命令の実行クロックを減少させ（ステップＳ６）、この後、ステップＳ３に戻る。また、ステップＳ３で、重なっていない場合（ＮＯ）は、ステップＳ５に進む。

【００５９】

次に、ステップＳ５で、重なっていない場合（ＮＯ）は、対象となっているＰＬ命令を現在の実行クロックでＤＴＣ命令列に配置する（ステップＳ７）。この

後、ＤＴＣ命令列に設定されていないＰＬ命令があるか否かを調べ（ステップＳ８）、設定されていないＰＬ命令がある場合（ＹＥＳ）は、ＤＴＣ命令列に設定されていないＰＬ命令の内、最も実行クロックの遅いＰＬ命令を対象として（ステップＳ９）、ステップＳ３に戻る。また、ステップＳ８で、設定されていないＰＬ命令がない場合（ＮＯ）、本フローは終了する。

【００６０】

次に、図１２は、コンパイラによって行われるＰＳ命令の配置アルゴリズムを示したフローチャートであり、図１２を用いて、ＰＳ命令の配置方法について説明する。なお、図１２の各フローで行われる処理は、特に明記しない限りコンパイラによって行われる。

図１２において、まず最初に、プロセッサＣＰＵｍのストア命令実行クロックから１クロックを加えたクロックを、各ＰＳ命令の仮の実行クロックに設定する（ステップＳ１１）。次に、実行クロックの最も早い、すなわち最初に実行されるＰＳ命令を対象にする（ステップＳ１２）。

【００６１】

この後、対象となっているＰＳ命令が、すでにＤＴＣ命令列に配置されているＬＡＣ命令又はＳＡＣ命令と重なっているか否かを調べ（ステップＳ１３）、重なっている場合（ＹＥＳ）は、対象となっているＰＳ命令をＬＡＣ命令及びＳＡＣ命令と重ならないクロックまで実行クロックを増加させる（ステップＳ１４）。次に、現在ＤＴＣ命令列に配置されているＰＬ命令のロード範囲及びＰＳ命令のストア範囲に、配置しようとしているＰＳ命令が重なるか否かを調べ（ステップＳ１５）、重なる場合（ＹＥＳ）は、重なったＰＬ命令のロード範囲外又は重なったＰＳ命令のストア範囲外になるまで対象となっているＰＳ命令の実行クロックを増加させ（ステップＳ１６）、この後、ステップＳ１３に戻る。また、ステップＳ１３で、重なっていない場合（ＮＯ）は、ステップＳ１５に進む。

【００６２】

次に、ステップＳ１５で、重ならない場合（ＮＯ）は、現在ＤＴＣ命令列に配置されているＰＬ命令又はＰＳ命令が、配置しようとしているＰＳ命令のストア範囲に重なるか否かを調べる（ステップＳ１７）。ステップＳ１７で、重なる場

合（ＹＥＳ）は、すでにＤＴＣ命令列に配置されたＰＬ命令又はＰＳ命令がストア範囲に重ならなくなるまで、配置しようとしているＰＳ命令の実行クロックを増加させ（ステップＳ１８）、ステップＳ１３に戻る。

【００６３】

また、ステップＳ１７で、重ならない場合（ＮＯ）は、対象となっているＰＳ命令を現在の実行クロックでＤＴＣ命令列に配置する（ステップＳ１９）。この後、ＤＴＣ命令列に設定されていないＰＳ命令があるか否かを調べ（ステップＳ２０）、設定されていないＰＳ命令がある場合（ＹＥＳ）は、ＤＴＣ命令列に設定されていないＰＳ命令の内、最も実行クロックの早いＰＳ命令を対象として（ステップＳ２１）、ステップＳ１３に戻る。また、ステップＳ２０で、設定されていないＰＳ命令がない場合（ＮＯ）、本フローは終了する。

このようにして、コンパイラは、ＤＴＣ命令列への各ＰＬ命令及び各ＰＳ命令の配置を行う。

【００６４】

図１３は、上記のようなアルゴリズムを用いてコンパイラで生成されたＤＴＣ命令列の例を示した図である。なお、図１３において、ＣＰＵ命令列における、ＬＷがロード命令を、ＳＷがストア命令をそれぞれ示している。また、集中共有メモリＣＳＭに対するアクセス（図１３では、メモリアクセスと示す）において、「PreLoad from Mem 32-35 to Cache 0-3」は、集中共有メモリＣＳＭのアドレス３２～３５のデータをキャッシュメモリ６のアドレス０～３にプレロードすることを示している。同様に、集中共有メモリＣＳＭに対するアクセスにおいて、「PostStore from Cache 0-3 to Mem 16-19」は、キャッシュメモリ６のアドレス０～３のデータを集中共有メモリＣＳＭのアドレス１６～１９にポストストアすることを示している。

【００６５】

このように、本実施の形態におけるキャッシュメモリシステム装置は、ハードウェア制御で動作する従来のキャッシュメモリシステム装置に、ソフトウェアで制御されるソフトウェア制御モードを追加すると共に、該ソフトウェア制御モード時にデータ転送を管理するプロセッサであるＤＴＣ８を追加し、該ＤＴＣ８に

対するプログラムの作成を行う際、キャッシュメモリ６のライン情報を管理するアルゴリズムをコンパイラに追加した。

【００６６】

このことから、ソフトウェア制御モード時において、ＤＴＣ８は、コンパイラの静的予測が可能な場合、コンパイラが生成したコードにしたがってプロセッサＣＰＵｍが必要とするデータをあらかじめキャッシュメモリ６に用意し、コンパイラの静的予測が不可能な場合、プロセッサＣＰＵｍからの情報を受け取ってアドレスを算出することにより動的にデータをキャッシュメモリ６に用意することから、キャッシュミスが発生せずにプロセッサに対して無同期実行が可能となる。

【００６７】

なお、上記実施の形態では、複数のプロセッサに対して無同期実行を行うマルチプロセッサシステムに使用する場合を例にして説明したが、本発明は、これに限定するものではなく、シングルプロセッサの場合においても適用することができ、キャッシュヒット率の改善及び処理時間の短縮等を図ることができる。

【００６８】

【発明の効果】

上記の説明から明らかなように、本発明のキャッシュメモリシステム装置によれば、ソフトウェア制御を行うソフトキャッシュ制御部と、ハードウェア制御を行うハードキャッシュ制御部とを備え、プロセッサは、ソフトキャッシュ制御部に対してソフトウェア制御を実行させ、例えばキャッシュミス等が発生してソフトウェア制御が実行不可能になると、ハードキャッシュ制御部に対してハードウェア制御を行わせるようにした。このことから、ある区間内で１００％のキャッシュヒット率を得ることができ、複数のプロセッサに対して無同期実行を行うマルチプロセッサシステムを実現することができる。

【００６９】

また、ソフトキャッシュ制御部は、コンパイラの静的予測によって生成されたコードにしたがって所望のデータをキャッシュメモリ内に格納するようにした。このことから、ある区間内で１００％のキャッシュヒット率を得ることができる

【００７０】

具体的には、ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行する前に、該データ読み出し命令で指定されるメインメモリのアドレスのデータを読み出してキャッシュメモリに格納するようにした。このことから、プロセッサによるロード命令等のデータ読み出し命令実行時に１００％のキャッシュヒット率を得ることができる。

【００７１】

更に、ソフトキャッシュ制御部は、メインメモリの所望のデータを読み出すデータ読み出し命令をプロセッサが実行すると同時に該データ読み出し命令で指定されたメインメモリのアドレスのデータをキャッシュメモリからプロセッサに転送させるようにした。このことから、プロセッサによるロード命令等のデータ読み出し命令実行時における処理時間の短縮を図ることができる。

【００７２】

また、具体的には、ソフトキャッシュ制御部は、メインメモリにデータを書き込むデータ書き込み命令をプロセッサが実行する前に、キャッシュメモリに対してプロセッサからのデータを格納するアドレスを指定するようにした。このことから、プロセッサによるストア命令等のデータ書き込み命令実行時におけるキャッシュメモリへのデータ書き込みの失敗をなくすことができる。

【００７３】

更に、ソフトキャッシュ制御部は、データ書き込み命令実行時にプロセッサからキャッシュメモリの指定したアドレスに書き込まれたデータを、該データ書き込み命令で指定されたメインメモリのアドレスに書き込むようにした。このことから、プロセッサによるストア命令等のデータ書き込み命令実行時における処理時間の短縮を図ることができる。

【００７４】

一方、ハードキャッシュ制御部は、ハードウェア制御時に、複数ウェイのセットアソシエイティブ方式を用いてキャッシュメモリのライン管理を行うようにし、ソフトキャッシュ制御部は、ソフトウェア制御時に、該複数ウェイの内少なく

とも１ウェイをフルアソシエイティブ方式を用いてキャッシュメモリのライン管理を行うようにした。このことから、キャッシュメモリに対するアクセス速度を低下させることなくキャッシュメモリの利用効率を高めることができる。

【００７５】

また具体的には、ソフトキャッシュ制御部を、キャッシュメモリに対するデータ転送制御を行う転送制御用プロセッサで構成するようにした。このことから、ソフトウェア制御時に、あらかじめプログラムされたソフトウェアにしたがってキャッシュメモリに対するデータ転送制御を行うことができる。

【図面の簡単な説明】

【図１】　本発明の実施の形態におけるキャッシュメモリシステム装置を使用するマルチプロセッサシステムの例を示した概略のブロック図である。

【図２】　本発明の実施の形態におけるキャッシュメモリシステム装置の例を示した概略のブロック図である。

【図３】　４ウェイ・セットアソシエイティブ方式を用いた場合におけるタグの対応表の例を示した図である。

【図４】　図３で示したタグの対応表のソフトウェア制御時における例を示した図である。

【図５】　図４で示したタグの対応表のハードウェア制御移行時における例を示した図である。

【図６】　図２のキャッシュメモリシステム装置のプレロード実行時における各部の動作例を示したブロック図である。

【図７】　ハードウェア制御時にキャッシュミスした場合の、図２のキャッシュメモリシステム装置における各部の動作例を示したブロック図である。

【図８】　ハードウェア制御時にキャッシュヒットした場合の、図２のキャッシュメモリシステム装置における各部の動作例を示したブロック図である。

【図９】　図２のキャッシュメモリシステム装置のポストストア実行時における各部の動作例を示したブロック図である。

【図１０】　コンパイラによって行われるプロセッサＣＰＵｍに対するアセンブラコード及びＤＴＣコードの生成過程を示したブロック図である。

【図１１】　コンパイラによって行われるＰＬ命令の配置アルゴリズムを示したフローチャートである。

【図１２】　コンパイラによって行われるＰＳ命令の配置アルゴリズムを示したフローチャートである。

【図１３】　コンパイラで生成されたＤＴＣ命令列の例を示した図である。

【図１４】　従来のキャッシュメモリシステム装置の構成例を示した概略のブロック図である。

【符号の説明】

１　キャッシュメモリシステム装置

５　リードバッファ

６　キャッシュメモリ

７　タグメモリ

８　ＤＴＣ（ソフトキャッシュ制御部）

９　ハードキャッシュ制御部

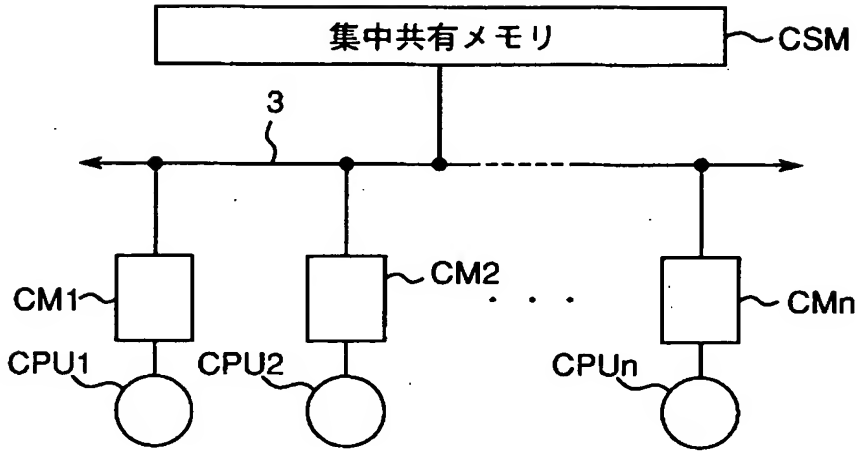
ＣＰＵ１～ＣＰＵｎ　プロセッサ

ＣＭ１～ＣＭｎ　キャッシュメモリ部

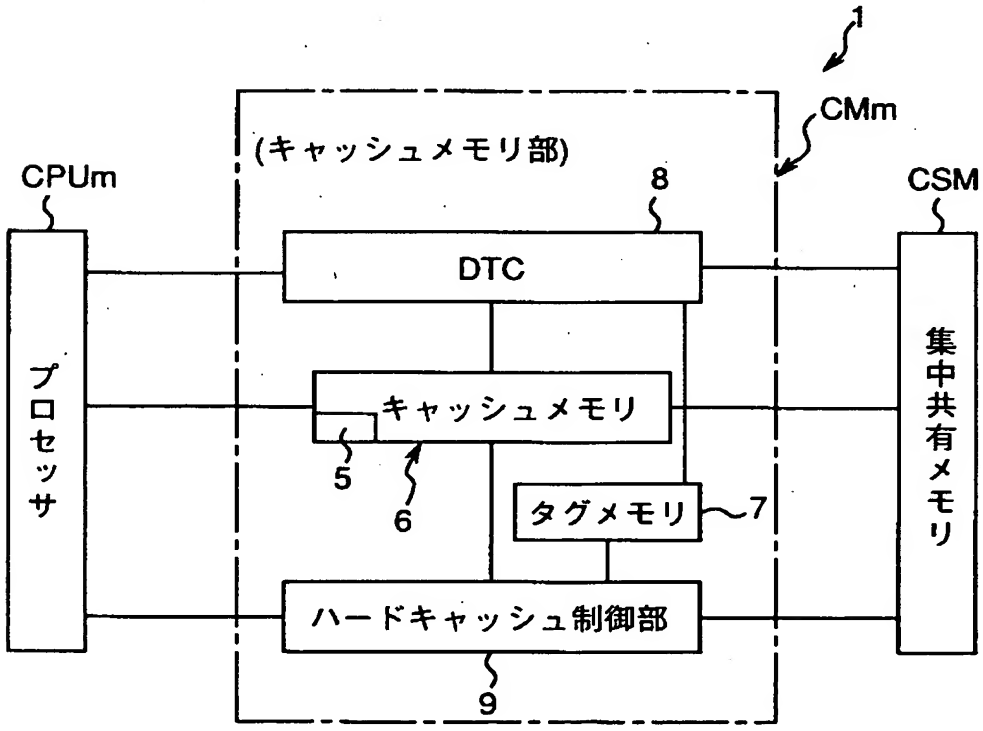
ＣＳＭ　集中共有メモリ（メインメモリ）

【書類名】　図面

【図１】


集中共有メモリ
CSM
3
CM1
CM2
CMn
CPU1
CPU2
CPUn

【図２】


1
CMm
(キャッシュメモリ部)
CPUm
8
CSM
DTC
プロセッサ
キャッシュメモリ
5
6
タグメモリ
7
集中共有メモリ
ハードキャッシュ制御部
9

【図３】

タグ	キー			
	第1ウェイ	第2ウェイ	第3ウェイ	第4ウェイ
0	16	24	—	—
1	01	09	17	25
2	18	10	02	—
3	11	43	19	27
4	20	28	—	—
5	05	13	21	29
6	14	22	—	—
7	07	23	31	—

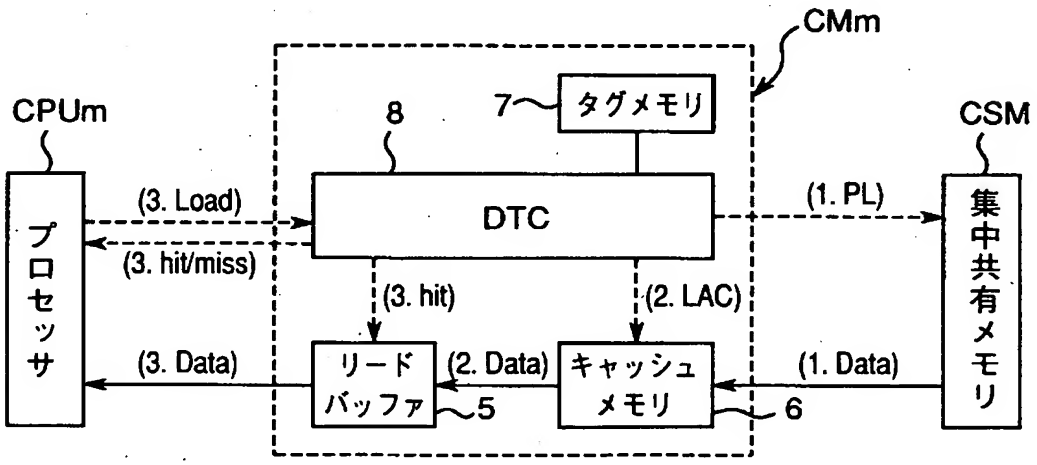
【図４】

タグ	キー			
	第1ウェイ	第2ウェイ	第3ウェイ	第4ウェイ
0	16	24	—	37
1	01	09	17	25
2	18	10	02	45
3	11	43	19	27
4	20	28	—	33
5	05	13	21	29
6	14	22	—	53
7	07	23	31	—

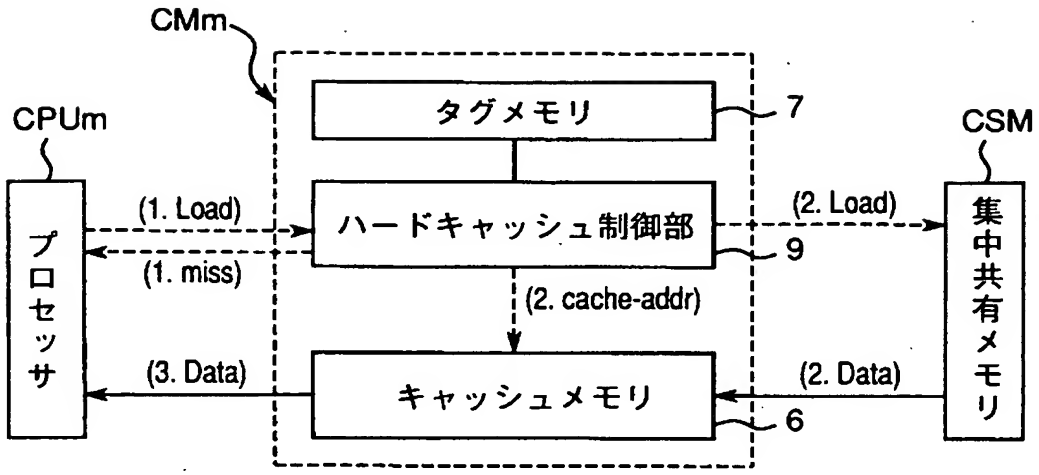
【図５】

タグ	キー			
	第1ウェイ	第2ウェイ	第3ウェイ	第4ウェイ
0	16	24	—	—
1	01	09	17	25
2	18	10	02	—
3	11	43	19	—
4	20	28	—	—
5	05	13	21	—
6	14	22	—	—
7	07	23	31	—

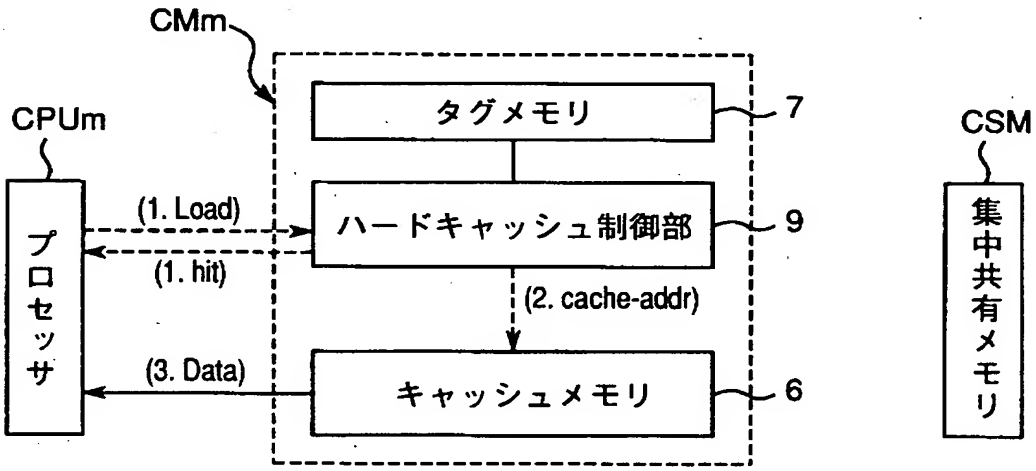
【図６】



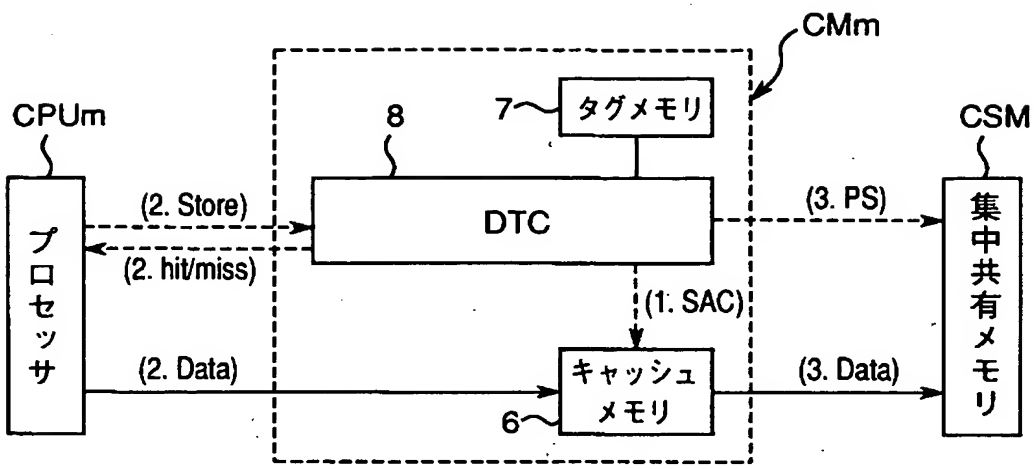
【図７】


CMm
CPUm
タグメモリ
7
CSM
プロセッサ
(1. Load)
ハードキャッシュ制御部
(2. Load)
9
集中共有メモリ
(1. miss)
(2. cache-addr)
(3. Data)
キャッシュメモリ
(2. Data)
6

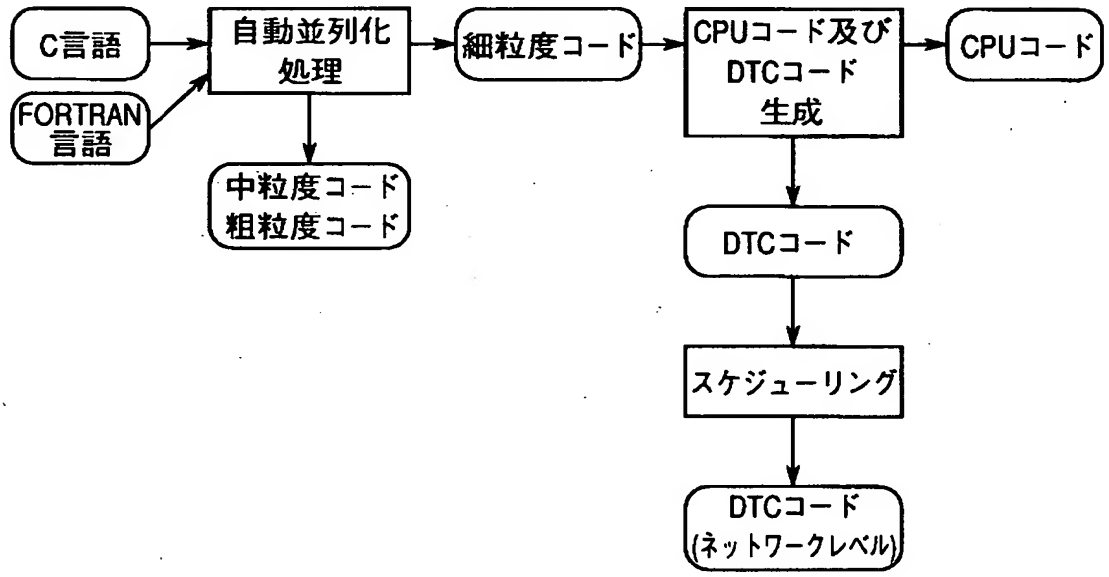
【図８】


CMm
CPUm
タグメモリ
7
CSM
プロセッサ
(1. Load)
ハードキャッシュ制御部
9
集中共有メモリ
(1. hit)
(2. cache-addr)
(3. Data)
キャッシュメモリ
6

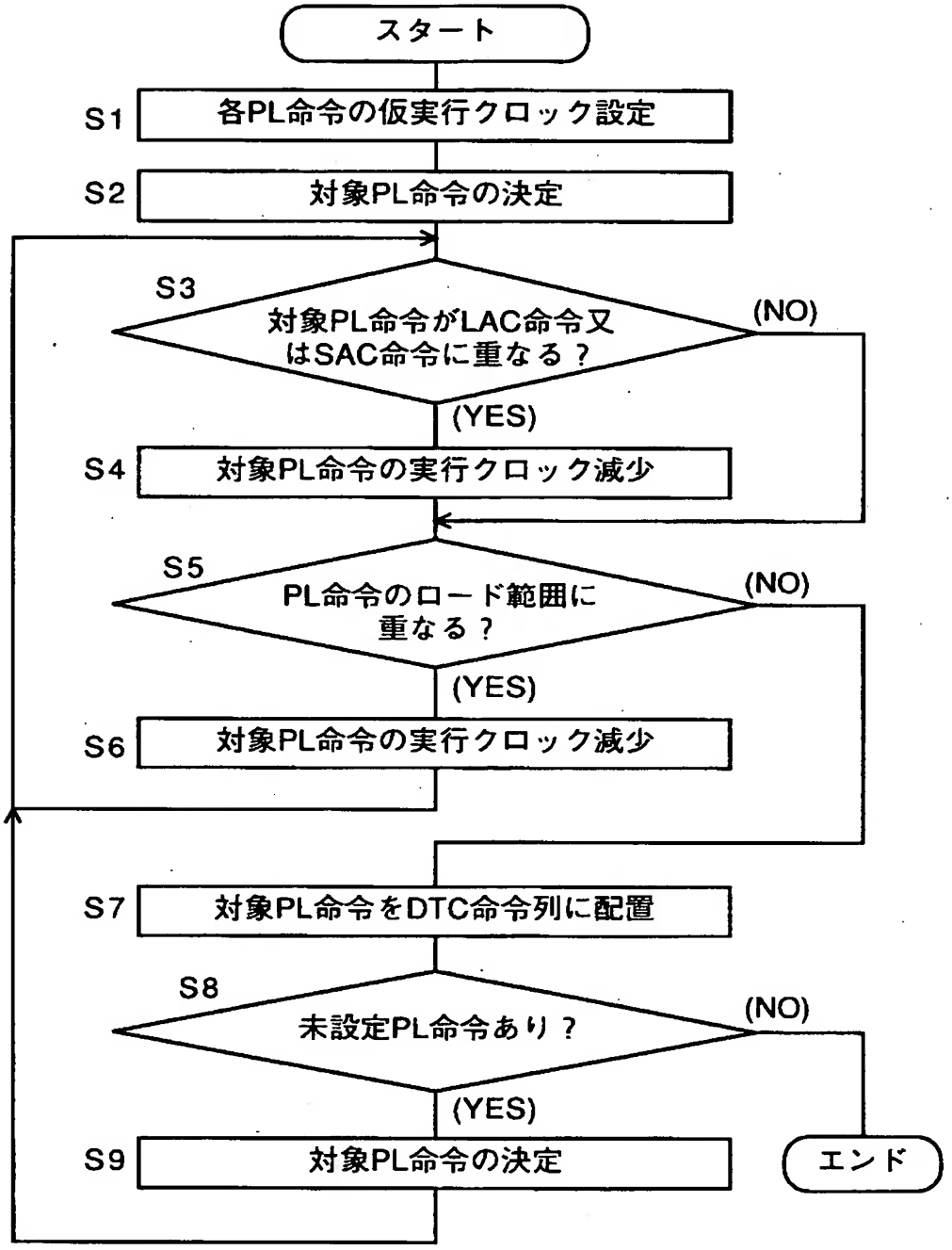
【図９】


CMm
CPUm
8
7
タグメモリ
CSM
プロセッサ
(2. Store)
(2. hit/miss)
DTC
(3. PS)
集中共有メモリ
(1. SAC)
(2. Data)
キャッシュメモリ
(3. Data)
6

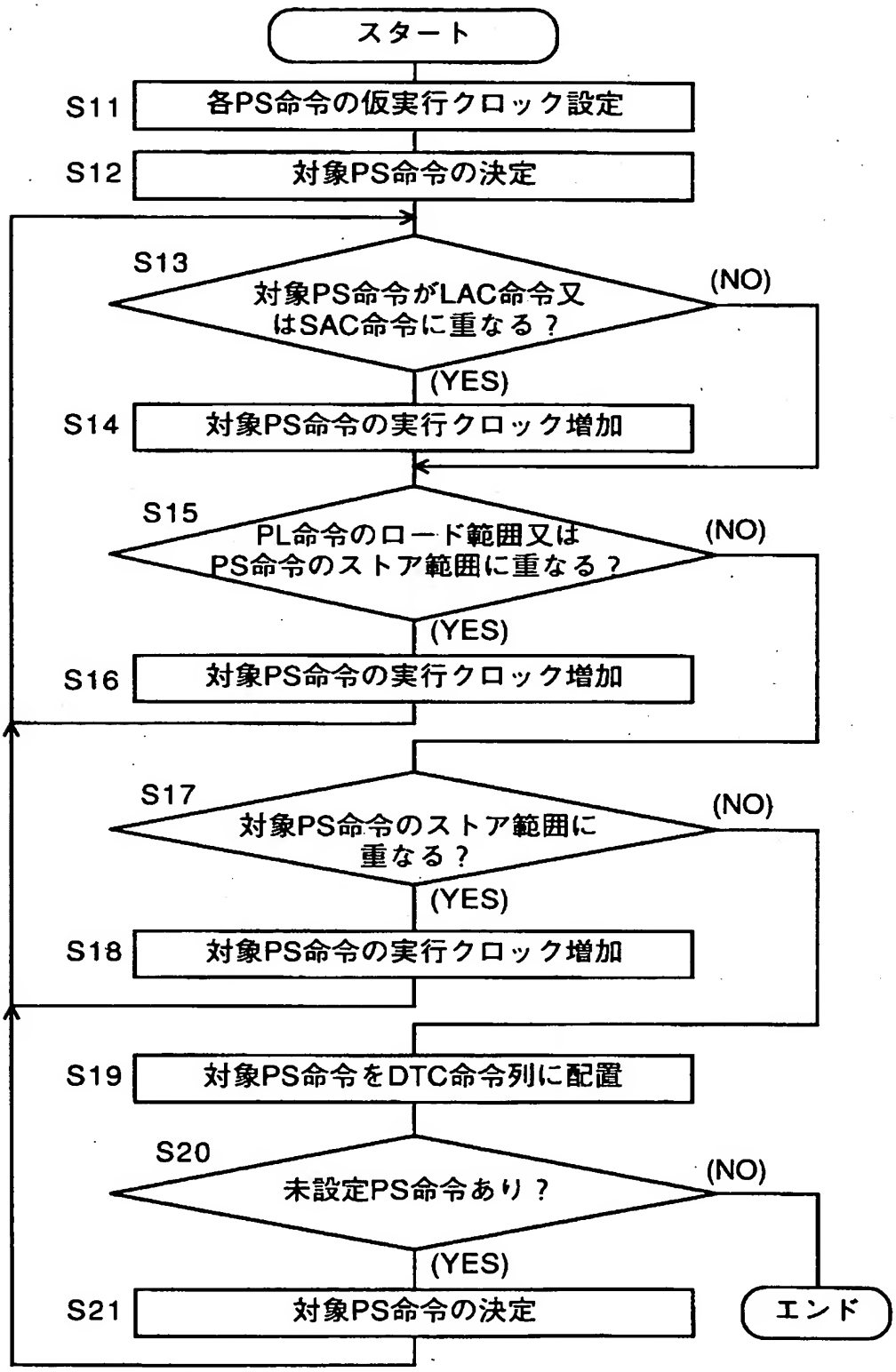
【図１０】


C言語
FORTRAN言語
自動並列化処理
中粒度コード
粗粒度コード
細粒度コード
CPUコード及びDTCコード生成
CPUコード
DTCコード
スケジューリング
DTCコード
(ネットワークレベル)

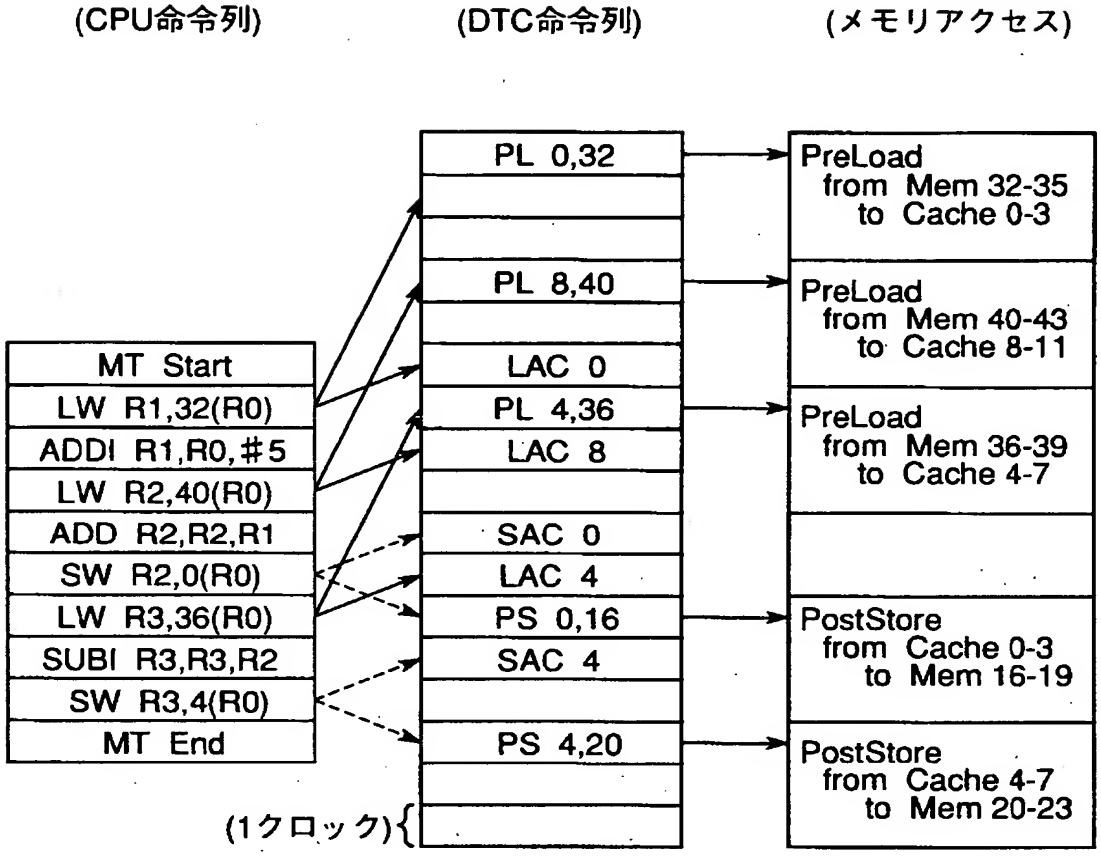
【図１１】


スタート
S1
各PL命令の仮実行クロック設定
S2
対象PL命令の決定
S3
対象PL命令がLAC命令又はSAC命令に重なる？
(NO)
(YES)
S4
対象PL命令の実行クロック減少
S5
PL命令のロード範囲に重なる？
(NO)
(YES)
S6
対象PL命令の実行クロック減少
S7
対象PL命令をDTC命令列に配置
S8
未設定PL命令あり？
(NO)
(YES)
S9
対象PL命令の決定
エンド

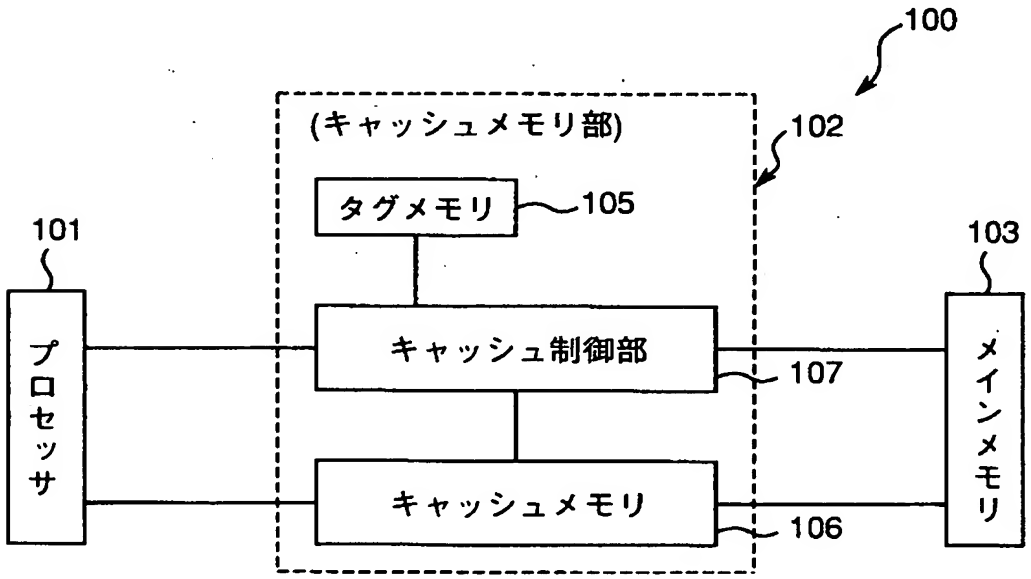
【図１２】


スタート
S11
各PS命令の仮実行クロック設定
S12
対象PS命令の決定
S13
対象PS命令がLAC命令又はSAC命令に重なる？
(NO)
(YES)
S14
対象PS命令の実行クロック増加
S15
PL命令のロード範囲又はPS命令のストア範囲に重なる？
(NO)
(YES)
S16
対象PS命令の実行クロック増加
S17
対象PS命令のストア範囲に重なる？
(NO)
(YES)
S18
対象PS命令の実行クロック増加
S19
対象PS命令をDTC命令列に配置
S20
未設定PS命令あり？
(NO)
(YES)
S21
対象PS命令の決定
エンド

【図１３】


(CPU命令列)
(DTC命令列)
(メモリアクセス)
MT Start
LW R1,32(R0)
ADDI R1,R0,#5
LW R2,40(R0)
ADD R2,R2,R1
SW R2,0(R0)
LW R3,36(R0)
SUBI R3,R3,R2
SW R3,4(R0)
MT End
(1クロック)
PL 0,32
PL 8,40
LAC 0
PL 4,36
LAC 8
SAC 0
LAC 4
PS 0,16
SAC 4
PS 4,20
PreLoad from Mem 32-35 to Cache 0-3
PreLoad from Mem 40-43 to Cache 8-11
PreLoad from Mem 36-39 to Cache 4-7
PostStore from Cache 0-3 to Mem 16-19
PostStore from Cache 4-7 to Mem 20-23

【図１４】


100
(キャッシュメモリ部)
102
タグメモリ
105
101
103
プロセッサ
キャッシュ制御部
107
メインメモリ
キャッシュメモリ
106

【書類名】　要約書

【要約】

【課題】　キャッシュミスが発生せずにマルチプロセッサシステム等における無同期実行を可能にするキャッシュメモリシステム装置を得る。

【解決手段】　ソフトウェアで制御されるソフトウェア制御モードを追加すると共に、該ソフトウェア制御モード時にデータ転送を管理するプロセッサであるＤＴＣ８を追加し、該ＤＴＣ８に対するプログラムの作成を行う際、キャッシュメモリ６のライン情報を管理するアルゴリズムをコンパイラに追加し、ソフトウェア制御モード時において、ＤＴＣ８は、コンパイラの静的予測が可能な場合、コンパイラが生成したコードにしたがってプロセッサＣＰＵｍが必要とするデータをあらかじめキャッシュメモリ６に用意し、コンパイラの静的予測が不可能な場合、プロセッサＣＰＵｍからの情報を受け取ってアドレスを算出することにより動的にデータをキャッシュメモリ６に用意するようにした。

【選択図】　図２

出　願　人　履　歴　情　報

識別番号　　〔３９６０２３９９３〕

１．変更年月日　　１９９６年１０月２８日
［変更理由］　新規登録
住　所　　東京都港区新橋６丁目１６番１０号
氏　名　　株式会社半導体理工学研究センター

２．変更年月日　　２００１年　３月２３日
［変更理由］　住所変更
住　所　　神奈川県横浜市港北区新横浜３丁目１７番地２　友泉新横浜ビル６階
氏　名　　株式会社半導体理工学研究センター